Singularity

Galen Hunt
James Larus
David Tarditi

Microsoft Research

MSR TAB
June 21, 2005

General-purpose Computing Platform (PC)

- Software is difficult to install, maintain, and administer
- Applications interact in complex, unpredictable ways
- Almost no users understand computers or software and so react naively to unexpected behavior
- System administration is costly and unavailable to most

Talk Summary

- Advances in languages, compilers, and tools open the possibility of improving software
- Singularity uses these advances as a basis to build more reliable systems and applications
- Systems built on Singularity expand software delivery opportunities

Advances in Language & Compiler Technology

- Software advances enable better system architectures
  - more robust and verifiable than existing (40 year old) model

	Advance	
	Language safety	**Program and system verification**
Technology	expressive safe languages and type systems optimizing compilers end-to-end safety checking	explicit specifications and system descriptions defect detection/testing tools static program analysis
Benefits	increasingly precise analysis systems with finer-grain isolation and fault-containment boundaries	improved static and dynamic error detection strong system guarantees

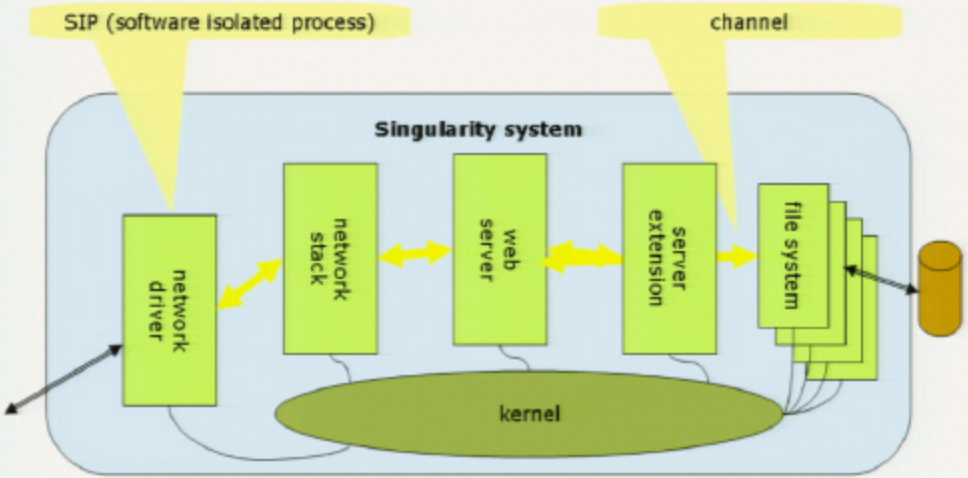
Singularity Project

- Develop technology and infrastructure to build more dependable software
  - language support to increase software quality
  - tools to ensure correct software behavior
  - OS architecture to enhance security, reliability, and effectiveness of tools
  - superior reliability, sufficient performance
- Large research project
  - current: ~dozen researchers & RSDEs
  - Redmond, SVC, Cambridge
- System running on hardware and Virtual PC
- Not Windows successor!

Key Aspects of Singularity

- Software-isolated processes (SIPs)
  - inexpensive isolation — memory, communications, failure — boundaries
  - OS manages and reclaims resources
  - "closed world" for program analysis
  - single isolation and extension model for all parts of application and system
- Merge OS and language runtime (VM/CLR)
  - prohibit unsafe code
  - remove duplicative APIs and security abstractions
  - fast, lightweight managed code run-time system
  - typed assembly language (TAL) reduces trusted computing base
- Language extensions to improve reliability
  - Spec#/Sing# specifications and verification
  - channel contracts
  - explicit and verified resource usage and reclamation

Singularity Architecture

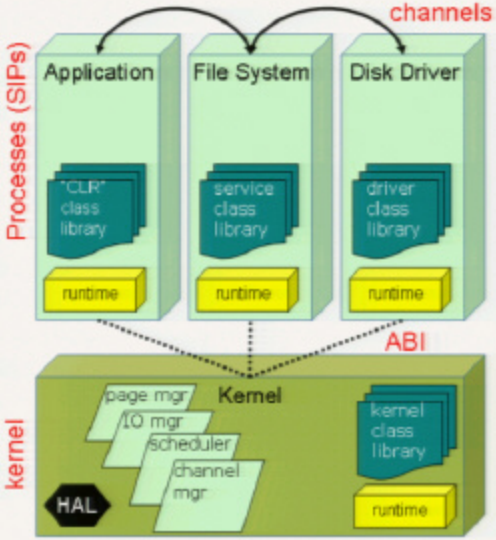


Presentation

- Introduction (Jim)
- Singularity OS (Galen)
- Language, compiler & runtime (David)
- Opportunities (Jim)

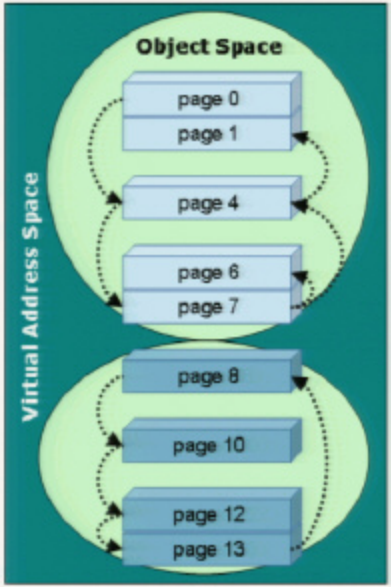
Detailed Architecture

- Microkernel
  - apps, extensions, services, and drivers are all processes
  - HAL w/ PIC, RTC, timer, and console output
- Closed processes (SIPs)
  - no shared memory
  - no dynamic code loading
  - no dynamic code generation
- IPC via channels w/ contracts
- Abstract instruction set
  - type safe, memory safe MSIL
  - all third-party code is safe
  - layer of indirection
- Well-defined, strongly-versioned application binary interface (ABI)



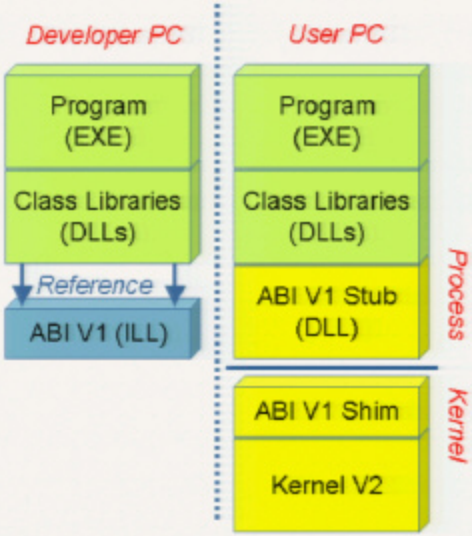
Software Isolated Processes (SIPs)

- Each SIP
  - contains verified user code
    - type safe
    - no privileged instructions
  - has exclusive ownership of pages
  - contains pointers to its pages only
    - self-contained for GC
  - has own runtime
- SIPs can
  - run in kernel address space
  - run in ring0
    - kernel hw protection domain



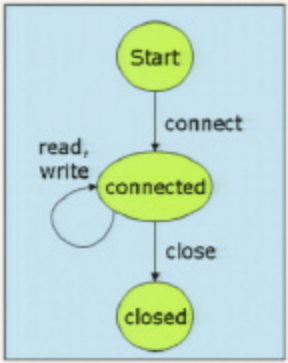
Application Binary Interface (ABI)

- Kernel/Process Boundary
  - strongly versioned
  - procedural ABI
  - transitions GC domain
  - typically wrapped in BCL
- ABI calls affect one process only
  - cross-process operations exclusively through channels
- Language support to separate interface from implementation
- V1 ABI (163 functions)
  - 42: threading & synchronization
  - 36: debugging
  - 24: process hand-off
  - 19: segmented stacks
  - 18: runtime
  - 16: channels
  - 8: device I/O

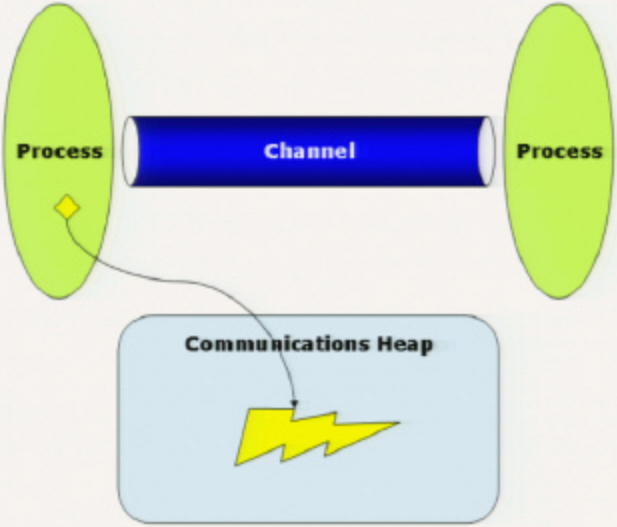


Channels

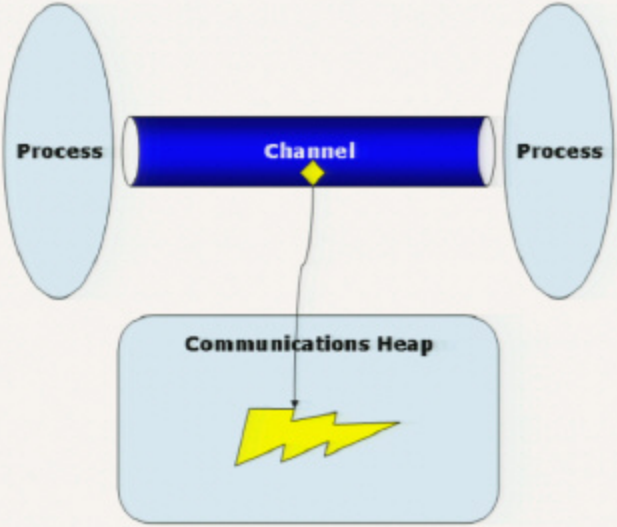
- Channels
  - asynchronous & bi-directional
  - exactly two endpoints
  - strongly typed with behavioral contract
    - format of messages
    - behavioral interface
- Messages may contain values and endpoints
- Messages and endpoints are linear
  - send removes message from sender
  - receive transfers message to receiver
  - at most one owning process at any time



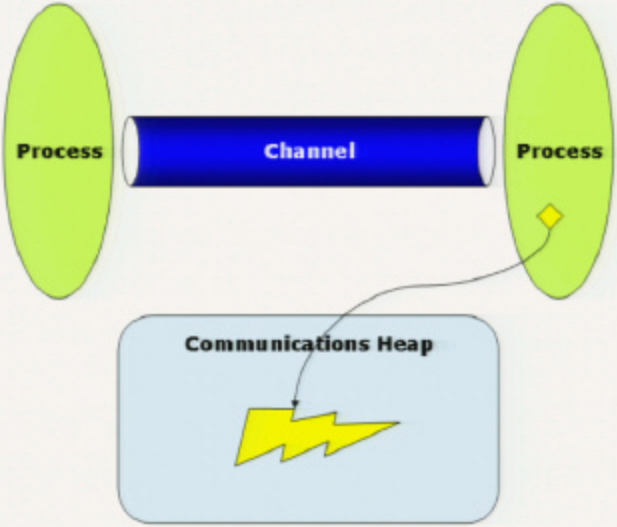
Channel Communication



Channel Communication



Channel Communication



Micro Performance

	Cost (CPU Cycles)			
	Singularity	**FreeBSD**	**Linux**	**Windows**
ABI call	37	861	547	443
thread yield	643	907	3,316	750
2 thread wait-set ping pong	2,928	8,168	15,034	3,419
2 message ping pong	3,849	15,815	23,618	(LPC) 4,653 (NP) 13,371
create and start process	218,236	807,581	658,911	7,231,038

Status (6/05)

- Bartok compiler and runtime
  - 5 GCs
  - lightweight runtime environment (~300K)
- Singularity kernel
  - processes and channels supported
  - drivers and services 'channelized'
- Sing# language support for communication and contracts
  - non-null checking
  - additional static checkers under development
- Non-trivial functionality
  - Microsoft Cassini ASP.NET web server
  - nameserver and volume manager
  - Boxwood B-tree file system

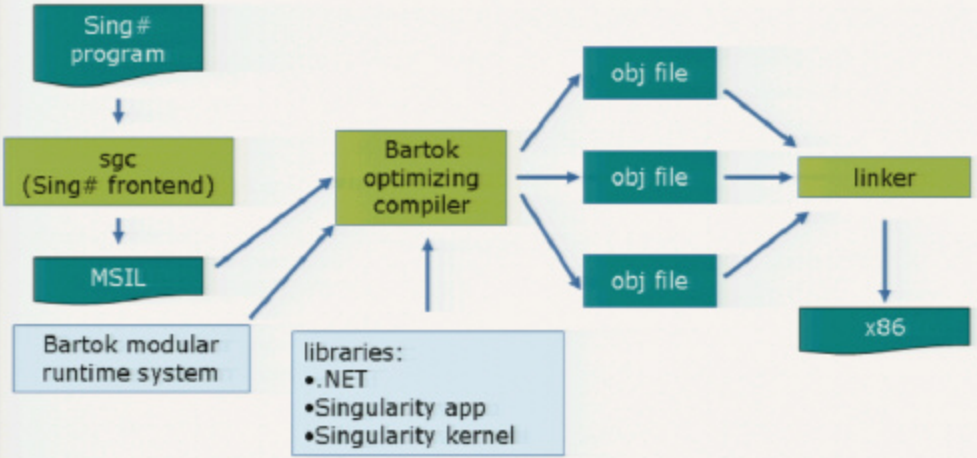
Presentation

- Introduction
- Singularity OS
- Language, compiler & runtime
- Opportunities

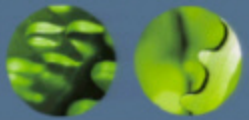
Singularity Language, Compiler, and Runtime

- Systems programming in safe, garbage-collected language
- Built on Bartok compiler & run-time system and Spec# language
- Language extensions
  - for systems programming
  - for program specification and verification
- Highly optimizing, ahead-of-time compiler
- Small, customizable run-time system
  - support distinct runtimes in separate processes

Compilation

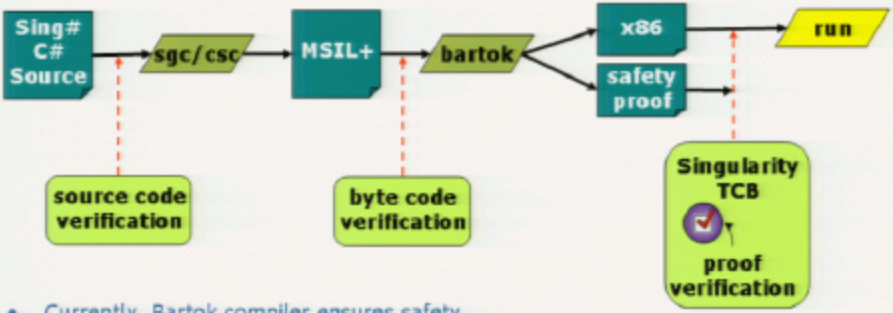


Bartok targets Singularity and Windows



Safety Verification

- System integrity depends on code safety
  - type and memory safety underlies SIP isolation
- Detect errors early, trust late



- Currently, Bartok compiler ensures safety
  - part of Singularity's trusted computing base (TCB)
- Working on using typed assembly language (TAL) to reduce TCB
  - allow other compilers
  - greatly reduce TCB size and complexity

C# Language Features

Supported:

- Core types + object model instructions
  - integers, floats, objects, managed pointers, unmanaged pointers
- Garbage collection, including finalizers, weak references
- Exception handling
- Lazy type initialization
- Delegates
- Unsafe code (kernel, runtime use only)
- Data layout attributes
- Generics (coming soon)

Not Supported:

- Dynamic class loading
- Reflection: very limited support
- Code access security
- Platform invoke

Sing# Extensions for Systems Programming

- Sing# based on Spec# (previous TAB)
- Channel contracts
  - specify typed message passing and valid protocol sequences
  - efficient implementation based on pre-allocated buffers
- rep structs are hierarchical structures exchanged over channels
- Custom heaps provide explicit, compiler verified, resource management for endpoints and exchangeable data
- switch-receive statement for asynchronous event pattern matching
- Linear data structures explicitly manage resources
- Overlays for type-safe structural casts of arrays of scalars

Channel Support

- Problem: efficient communications without data sharing
- Answer: "snail-mail" semantics
  - data transferred from sender to receiver
  - sender cannot retain message (can keep copy)
  - permits copying or pointer passing
- Enforced by type system
- Designate special types as exchangeable
  - primitive types
  - pointers to rep structs
  - pointers to vectors of rep structs
- rep structs are like structs except:
  - allocated outside GC heap
  - cannot point into GC heap (avoid sharing)
  - cannot be used after being sent in message
- Rep structs and primitive types are only types passed between processes

Contract and Rep Struct

```
namespace Sample {
   public rep struct Info {
      public int NumLookups;
      public int NumAdds;
   }
   public contract InfoContract : ServiceContract {
      out message Ready();
      in message GetInfo(int id);
      out  message SendInfo(Info * stats);

      override state Start : Ready! -> ReadyState;
      state ReadyState : GetInfo? -> SendInfo! -> ReadyState;
   }
}
```

Base contract

Message sent by contract exporter

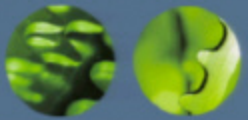
Message received

Message sent

Next state

Channel Contracts

- Contract declares
  - message types (name, argument types, direction)
  - state machine defining all valid message sequences
- Provides
  - channel endpoint types: Imp, Exp
  - channel construction method
  - typed send and receive methods
- Efficient
  - pre-allocated buffers
  - state machine and finite outstanding messages



Contract and Rep Struct

```
namespace Sample {
   public rep struct Info {
      public int NumLookups;
      public int NumAdds;
   }
   public contract InfoContract : ServiceContract {
      out message Ready();
      in message GetInfo(int id);
      out  message SendInfo(Info * stats);

      override state Start : Ready! -> ReadyState;
      state ReadyState : GetInfo? -> SendInfo! -> ReadyState;
   }
}
```

Base contract

Message sent by contract exporter

Message received

Message sent

Next state

Optimizing Compiler

- Bartok supports whole program and separate compilation
  - sealed processes allow whole program optimization
  - predecessor (Marmot) demonstrated whole program performance competitive with C/C++
- Assists specialization of runtime systems
  - remove unused or disabled language features
  - tree-shaking eliminates unused classes/methods/fields
- Extensive optimization
  - 30+ optimizations: high-level, medium-level, code generation, runtime focused

Multiple Run-Time Environments

- Each process's run-time environment is autonomous
  - libraries, garbage collector, run-time system, data representations specific to process
  - no cross-process pointers and rep types
- Customize environment for its code
  - GC algorithm
    - real-time for responsiveness
    - generational for speed
  - constrained library enforce architecture and facilitate analysis
    - e.g., no socket library for device drivers
  - insert additional run-time tests for less trusted code
    - e.g., security automata

Dynamic Memory Usage: Hello World

	Virtual address size (bytes)			
	Singularity	FreeBSD	Linux	Windows
C w/ shared lib		1,192K	2,624K	737K
C w/o shared lib		232K	1,788K	663K
C++ w/ shared lib		2,124K	4,216K	1,115K
C++ w/o shared lib		700K	2,372K	704K
C# w/ GC	316K			3,750K

Presentation

- Introduction
- Singularity OS
- Language, compiler & runtime
- Opportunities

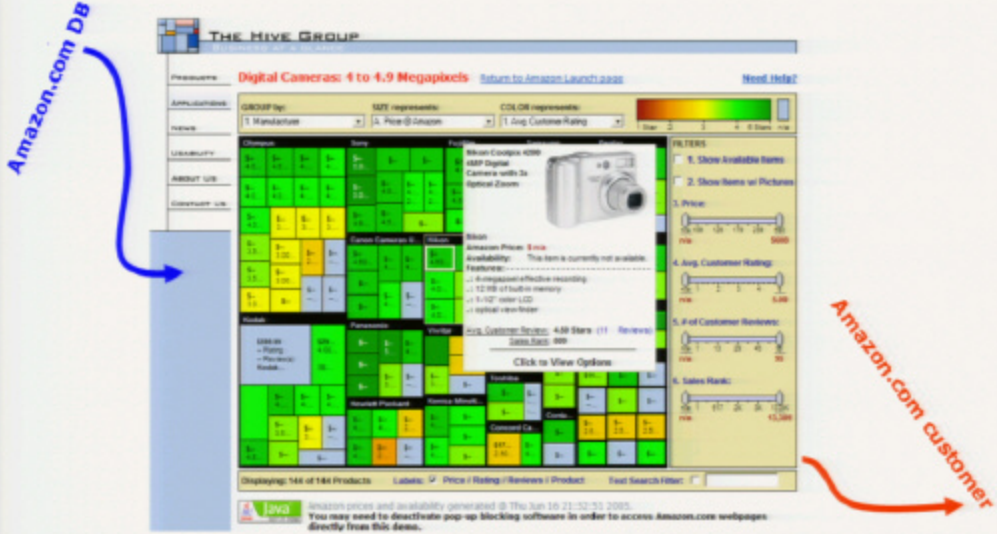
Software as a Service

- Free users from system administration
- Deliver and remotely support software
  - $29.99/yr to support Office on your home machines?
- Singularity
  - strong isolation
    - processes cannot interfere with each other
    - system has explicit and total control over communication mechanism
  - safe execution environment
    - OS controls verification, compilation, and run-time libraries
  - remotely administrable
    - OS controls installation and configuration
    - detect conflicts before execution
  - security model
    - identify and protect application resources (files, metadata)

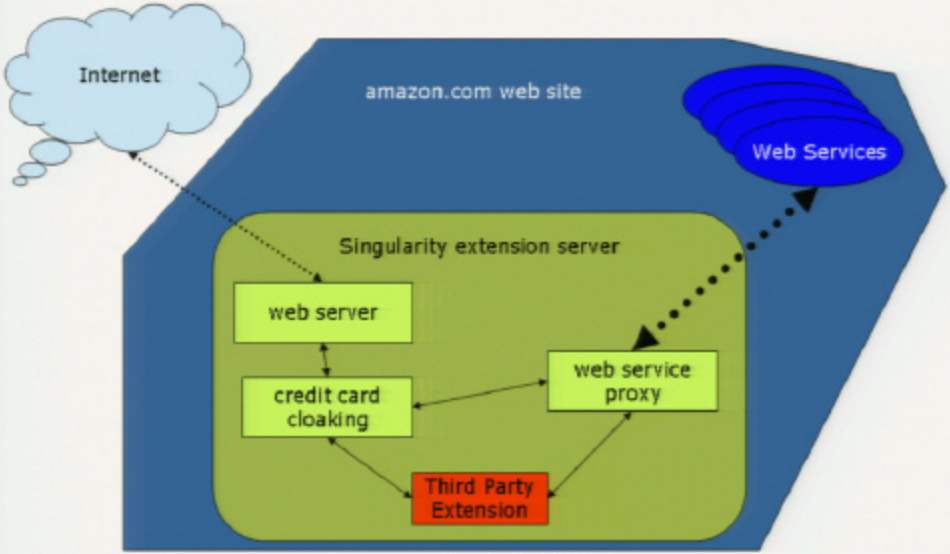
Web Extensions

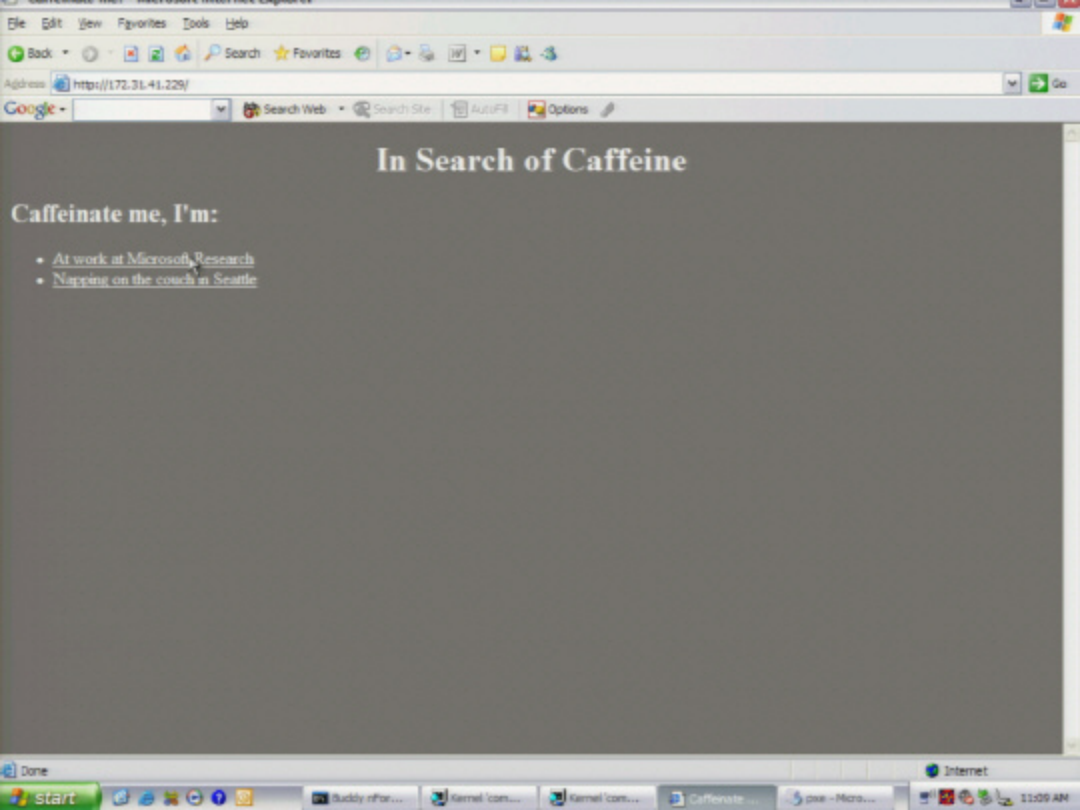
- Developers cannot anticipate or satisfy all uses of their software
  - non-trivial software provides rich extension mechanisms (COM, VBA, …)
- No safe way to host extensions to web site
  - e.g. Amazon.com storefronts or eBay bidding agent
- Developer with new idea must operate web site
- Singularity provides platform to safely host extensions
  - strong isolation
    - …
  - safe execution environment
    - …
  - remotely administrable
    - …
  - security model
    - …

Example


Amazon.com DB
THE HIVE GROUP
Digital Cameras: 4 to 4.9 Megapixels
Return to Amazon Launch page
Need Help?
GROUP by:
SIZE represents:
COLOR represents:
FILTERS
1. Show Available Items
2. Show Items w/ Pictures
3. Price:
4. Avg. Customer Rating:
5. # of Customer Reviews:
6. Sales Rank:
Click to View Options
Displaying: 144 of 144 Products
Amazon prices and availability generated @ Thu Jun 16 21:52:51 2005.
You may need to deactivate pop-up blocking software in order to access Amazon.com webpages directly from this demo.
Amazon.com customer

Safe Extensions to Web Sites




File Edit View Favorites Tools Help
Back Search Favorites
Address http://172.31.41.229/
Google Search Web Search Site AutoFill Options
In Search of Caffeine
Caffeinate me, I'm:
At work at Microsoft Research
Napping on the couch in Seattle
Done
Internet
start
11:09 AM

File Edit View Favorites Tools Help

Back Search Favorites

Address http://172.31.41.229/seattle_all.htm

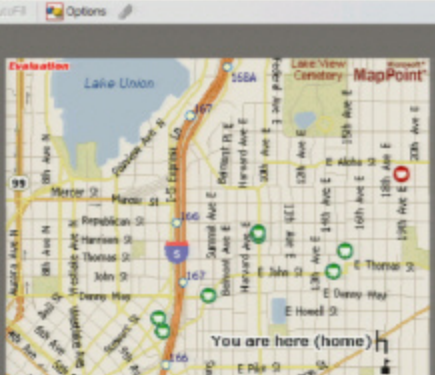
Google Search Web Search Site AutoFill Options

In Search of Caffeine

Looks like you have quite a few choices...

- Starbucks
- Seattle's Best Coffee
- Tully's
- Tully's with wireless

Take me to Redmond instead



http://172.31.41.229/seattle_sbc.htm

Internet

start

Buddy nFor... Kernel com... Kernel com... Caffeinate... pax - Micro...

11:10 AM

File Edit View Favorites Tools Help

Back Search Favorites

Address http://172.31.41.229/seattle_tullys.htm

Google Search Web Search Site AutoFill Options

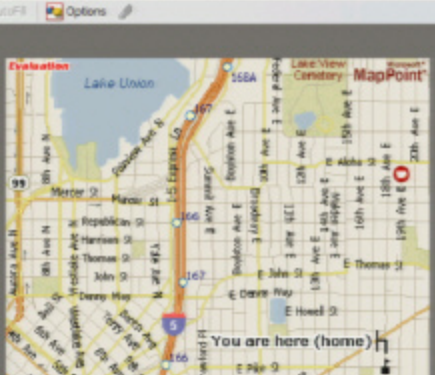
In Search of Caffeine

Looks like you have quite a few choices...

- Tully's
- Tully's with wireless

Show all chains

Take me to Redmond instead



http://172.31.41.229/redmond_all.htm

Internet

start

Buddy nFor... Kernel 'com... Kernel 'com... Caffeinate... pxe - Micro...

11:10 AM

Conclusion

- Singularity enables new systems and applications
  - started with: "what would a system that favored reliability over performance look like?"
  - building platform, tools, and technology to achieve this goal of improving software reliability
  - resulting system enables new software delivery vehicles and businesses
- Open research questions
  - verifiable programming abstractions
  - increased typed assembly language expressiveness
  - general error-handling and recovery mechanisms
  - more specification and better static defect detection tools
  - simpler, robust security model
  - first class application abstraction

Questions?

- http://singularity